Kotlin + Computer Science Mastery Guide

A Complete Learning Path for New Programmers

Phase 1: Kotlin Fundamentals + Programming Basics (Week 1-2)

Core Kotlin Concepts

- **Variables & Types**
  - `val` vs `var`
  - Type inference
  - Nullable types (`String?`)
- **Functions & Control Flow**
  - Function syntax
  - `when` expressions
  - Loops and conditions
- **Classes & Objects**
  - Basic classes
  - Data classes
  - Properties and methods

CS Fundamentals: Programming Concepts

- **Recursion** - Learn through Kotlin examples
- **Problem decomposition**
- **Code organization**

Practice Projects

1. Calculator with history
2. Number base converter
3. Simple text processor

Resources

- **Book**: "Kotlin Programming: The Big Nerd Ranch Guide"
- **Practice**: Kotlin Koans
- **CS Theory**: Watch Brian Harvey's Berkeley CS 61A (first 5 lectures)

Phase 2: Data Structures in Kotlin (Week 3-4)

Kotlin Collections Deep Dive

- **Lists**: `listOf()`, `mutableListOf()`
- **Sets**: `setOf()`, `mutableSetOf()`
- **Maps**: `mapOf()`, `mutableMapOf()`
- **Array operations**

CS Fundamentals: Data Structures

- **Stack** - Implement using Kotlin's `MutableList`
- **Queue** - Custom implementation
- **Linked Lists** - Build from scratch
- **Trees** - Binary trees in Kotlin
- **Graphs** - Adjacency lists using Maps

Kotlin-Specific Features

- Extension functions for collections
- Higher-order functions (`map`, `filter`, `reduce`)
- Sequences for lazy evaluation

Practice Projects

1. **Stack-based calculator**
2. **File system navigator** (tree structure)
3. **Social network analyzer** (graph algorithms)

```kotlin
// Example: Custom Stack in Kotlin
class Stack<T> {
    private val items = mutableListOf<T>()

    fun push(item: T) = items.add(item)
    fun pop(): T? = if (items.isEmpty()) null else items.removeAt(items.size - 1)
    fun peek(): T? = items.lastOrNull()
    val isEmpty: Boolean get() = items.isEmpty()
}
```

Resources

- **Book**: "The Algorithm Design Manual" by Steven Skiena

- **Videos**: Steven Skiena's Algorithm Lectures
- **Practice**: LeetCode Easy problems in Kotlin

Phase 3: Algorithms + Advanced Kotlin (Week 5-6)

Algorithm Categories

- **Sorting**: Implement QuickSort, MergeSort in Kotlin
- **Searching**: Binary search with Kotlin's type safety
- **Graph algorithms**: BFS, DFS using Kotlin coroutines
- **Dynamic Programming**: Memoization with Kotlin maps

Advanced Kotlin Features

- **Sealed classes** for algorithm state representation
- **Inline functions** for performance
- **Generics** for reusable algorithms
- **Operator overloading** for mathematical operations

Practice Projects

1. **Pathfinding visualizer**
2. **Sorting algorithm comparison tool**
3. **Text search engine** (with indexing)

```kotlin
// Example: Generic Binary Search
inline fun <T : Comparable<T>> List<T>.binarySearch(target: T): Int {
    var left = 0
    var right = size - 1

    while (left <= right) {
        val mid = (left + right) / 2
        when {
            this[mid] == target -> return mid
            this[mid] < target -> left = mid + 1
            else -> right = mid - 1
        }
    }
    return -1
}
```

Phase 4: Systems Programming with Kotlin (Week 7-8)

CS Fundamentals: Computer Architecture

- **Memory management** - Kotlin's garbage collection
- **CPU concepts** - How Kotlin/JVM utilizes processors
- **Storage hierarchy** - Caching strategies in Kotlin

Kotlin System Programming

- **File I/O** operations
- **Networking** with Kotlin coroutines
- **Multithreading** and concurrency
- **Performance optimization**

Practice Projects

1. **File backup system**
2. **Multi-threaded web scraper**
3. **Memory usage analyzer**

Resources

- **Book**: "Computer Systems: A Programmer's Perspective"
- **Videos**: Berkeley CS 61C lectures

Phase 5: Databases + Kotlin Integration (Week 9-10)

Database Fundamentals

- **Relational model**
- **SQL basics**
- **ACID properties**
- **Indexing and performance**

Kotlin Database Programming

- **SQLite with Kotlin**
- **Room database (Android)**
- **Exposed framework** for type-safe SQL
- **Connection pooling**

Practice Projects

1. **Personal finance tracker**
2. **Library management system**
3. **Data analysis tool**

kotlin

```kotlin
// Example: Type-safe database queries with Exposed
object Users : Table() {
    val id = integer("id").autoIncrement()
    val name = varchar("name", 50)
    val email = varchar("email", 100)

    override val primaryKey = PrimaryKey(id)
}

fun findUserByEmail(email: String): User? = transaction {
    Users.select { Users.email eq email }
        .singleOrNull()
        ?.let { User(it[Users.name], it[Users.email]) }
}
```

Phase 6: Networking + Distributed Systems (Week 11-12)

Network Programming Fundamentals

- **TCP/UDP protocols**
- **HTTP/HTTPS**
- **API design principles**
- **Serialization** (JSON, Protocol Buffers)

Kotlin Networking

- **Ktor framework** for servers and clients
- **Coroutines** for async networking
- **WebSocket implementation**
- **REST API development**

Distributed Systems Concepts

- **Microservices** with Kotlin
- **Load balancing**
- **Data consistency**

- **Fault tolerance**

Practice Projects

1. **Chat server and client**
2. **RESTful API with authentication**
3. **Distributed task scheduler**

Phase 7: Advanced Topics + Real-World Applications (Week 13-16)

Compiler Theory (Applied to Kotlin)

- **Lexical analysis**
- **Parsing techniques**
- **AST manipulation**
- **Kotlin compiler plugins**

Operating Systems Concepts

- **Process management**
- **Memory management**
- **File systems**
- **Concurrency control**

Advanced Kotlin Projects

1. **Domain-specific language (DSL)**
2. **Code analysis tool**
3. **Build system plugin**
4. **Performance monitoring library**

Mathematics for CS

- **Discrete mathematics**
- **Graph theory**
- **Probability and statistics**
- **Linear algebra** (for ML applications)

Comprehensive Project: Build a Complete System

Final Capstone Project Options

Option 1: Distributed Task Management System

- **Frontend**: Kotlin/JS or Android app
- **Backend**: Ktor microservices
- **Database**: PostgreSQL with Exposed
- **Features**: Real-time updates, load balancing, data analytics

Option 2: Code Analysis and Optimization Tool

- **Parser**: Custom Kotlin parser
- **Analysis**: Algorithm complexity analysis
- **Visualization**: Performance graphs
- **Integration**: IDE plugin development

Option 3: Educational Platform

- **Content**: Interactive CS lessons
- **Backend**: Kotlin server with WebSocket
- **Features**: Code execution sandbox, progress tracking
- **AI**: Simple recommendation system

Daily Study Schedule

Phase Structure (4 hours/day)

- **Hour 1**: Theory study (books/videos)
- **Hour 2**: Kotlin coding practice
- **Hour 3**: Project work
- **Hour 4**: CS problem solving

Weekly Goals

- **Monday-Wednesday**: New concepts
- **Thursday-Friday**: Implementation practice
- **Saturday**: Project integration
- **Sunday**: Review and reinforcement

Essential Resources Library

Books (Free/Open Source)

1. **"Structure and Interpretation of Computer Programs"** - Free online
2. **"Operating Systems: Three Easy Pieces"** - Free online
3. **"Mathematics for Computer Science"** - MIT OpenCourseWare
4. **"Designing Data-Intensive Applications"** - Martin Kleppmann

Video Courses

1. **Berkeley CS 61A** - Programming fundamentals
2. **MIT 6.042J** - Mathematics for CS
3. **Stanford CS 144** - Computer Networking
4. **Berkeley CS 186** - Database Systems

Practice Platforms

1. **LeetCode** - Algorithm practice
2. **HackerRank Kotlin** - Language-specific problems
3. **Kotlin Koans** - Interactive learning
4. **Project Euler** - Mathematical programming

Documentation & References

1. **Kotlin Official Documentation**
2. **Kotlin Language Specification**
3. **JVM Specification**
4. **Computer Science papers** (Google Scholar)

Assessment Milestones

Week 4 Checkpoint

- [ ] Implement all basic data structures in Kotlin
- [ ] Solve 20 easy algorithm problems
- [ ] Build a command-line application with persistence

Week 8 Checkpoint

- [ ] Create a multi-threaded application
- [ ] Implement 3 different sorting algorithms
- [ ] Build a simple database-backed web service

Week 12 Checkpoint

- [ ] Deploy a distributed system

- [ ] Optimize code for performance
- [ ] Demonstrate understanding of system design

Final Assessment

- [ ] Complete capstone project
- [ ] Technical presentation
- [ ] Code review with experienced developer
- [ ] Contribute to open-source Kotlin project

Career Preparation

Portfolio Projects Showcase

1. **Algorithm Visualizer** - Demonstrates CS knowledge
2. **Full-Stack Application** - Shows practical skills
3. **Performance Optimization** - Proves systems understanding
4. **Open Source Contribution** - Community involvement

Technical Interview Preparation

- **Data structures and algorithms** in Kotlin
- **System design** questions
- **Kotlin-specific** technical questions
- **Project walkthrough** preparation

Next Steps After Mastery

- **Android development** specialization
- **Backend/Server-side** Kotlin focus
- **Kotlin Multiplatform** for cross-platform development
- **Contribute to Kotlin ecosystem**

This guide combines practical Kotlin programming with solid computer science fundamentals. Each phase builds upon the previous one, ensuring both language mastery and CS understanding. Adjust the timeline based on your learning pace, but maintain the progression order for optimal learning.